

#3ed
11-18-02

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Araya Yasuteru, et al.	Examiner:	Unassigned
Serial No:	Unassigned	Art Unit:	Unassigned
Filed:	Herewith	Docket:	14669
For:	BUS PERFORMANCE EVALUATION METHOD FOR ALGORITHM DESCRIPTION	Dated:	June 1, 2001

Assistant Commissioner for Patents
Washington, D.C. 20231


11000 U.S. PTO
09/872091
06/01/01

CLAIM OF PRIORITY

Sir:

Applicant in the above-identified application hereby claims the right of priority in connection with Title 35 U.S.C. § 119 and in support thereof, herewith submits a certified copy of Japanese Patent Application No. 2000-166630, filed on June 2, 2000.

Respectfully submitted,


Paul J. Esatto, Jr.
Registration No.: 30,749

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343


CERTIFICATE OF MAILING BY "EXPRESS MAIL"

Express Mailing Label No.: EL 915257271 US

Date of Deposit: June 1, 2001

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents and Trademarks, Washington, D.C. 20231 on June 1, 2001.

Dated: June 1, 2001


Janet Grossman

OSP-10591-45

日 本 国 特 許 庁

PATENT OFFICE
JAPANESE GOVERNMENT

11000 U.S. PRO
09/872091



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 6月 2日

出 願 番 号

Application Number:

特願2000-166630

出 願 人

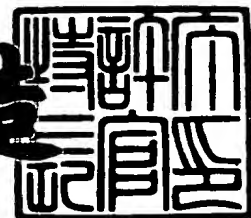
Applicant (s):

日本電気アイシーマイコンシステム株式会社

2001年 3月 2日

特許庁長官
Commissioner,
Patent Office

及 川 耕 造



出証番号 出証特2001-3013380

【書類名】 特許願

【整理番号】 01211132

【提出日】 平成12年 6月 2日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 17/50

【発明の名称】 アルゴリズム記述におけるバスの性能評価方法

【請求項の数】 8

【発明者】

 【住所又は居所】 神奈川県川崎市中原区小杉町一丁目4 0 3 番 5 3 日本
電気アイシーマイコンシステム株式会社内

 【氏名】 荒谷 泰輝

【発明者】

 【住所又は居所】 神奈川県川崎市中原区小杉町一丁目4 0 3 番 5 3 日本
電気アイシーマイコンシステム株式会社内

 【氏名】 丸山 勇一

【特許出願人】

 【識別番号】 000232036

 【氏名又は名称】 日本電気アイシーマイコンシステム株式会社

【代理人】

 【識別番号】 100108578

 【弁理士】

 【氏名又は名称】 高橋 詔男

【代理人】

 【識別番号】 100064908

 【弁理士】

 【氏名又は名称】 志賀 正武

【選任した代理人】

 【識別番号】 100101465

 【弁理士】

【氏名又は名称】 青山 正和

【選任した代理人】

【識別番号】 100108453

【弁理士】

【氏名又は名称】 村山 靖彦

【手数料の表示】

【予納台帳番号】 008707

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9901153

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 アルゴリズム記述におけるバスの性能評価方法

【特許請求の範囲】

【請求項 1】 アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、

前記ソースを、前記バスにデータ転送が生じた際に、特定の評価関数を実行するように修正することにより、アーキテクチャ設計を行うためのシミュレーションプラットフォームを構築することを特徴とするアルゴリズム記述における性能評価方法。

【請求項 2】 前記シミュレーションプラットフォームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、前記バスの性能評価を上位設計の段階において行うことを特徴とする請求項 1 に記載のアルゴリズム記述における性能評価方法。

【請求項 3】 前記評価対象となるバスの性能評価結果を前記アルゴリズム検証で用いたソースにフィードバックすることにより、上位設計の段階において、アーキテクチャ設計を行うことを特徴とする請求項 1 または 2 に記載のアルゴリズム記述における性能評価方法。

【請求項 4】 アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、

前記ソースを用いてアーキテクチャ設計を行うためのシミュレーションプラットフォームを構築し、

前記構築したシミュレーションプラットフォームを用いて評価対象となるバスの性能評価を行い、その評価結果に基づき前記ソースを修正して前記シミュレーションプラットフォームを再構築し、

前記再構築されたシミュレーションプラットフォームを用いて前記性能評価を行うことを特徴とするアルゴリズム記述におけるバスの性能評価方法。

【請求項 5】 前記ソースを用いてアーキテクチャ設計を行うためのシミュレーションプラットフォームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、前記バスの性能評価を上位設計の段階において行うことを特徴とする請求項 1 に記載のアルゴリズム記述における性能評価方法。

ュレーションプラットフォームを構築するために、

前記ソースを所定の単位で解析することによりハードウェアとソフトウェア記述に分け、

前記バスのトラフィックをカウントする評価関数を作成し、
前記バスを使用したデータ転送が発生する都度、前記作成された評価関数を実行するように前記ソース構文を修正することを特徴とする請求項4に記載のアルゴリズム記述におけるバスの性能評価方法。

【請求項6】 前記性能評価を行うために、

前記評価関数に基づいて性能評価を行い、

最終的に算出された処理レートに対するバストラフィックにより、前記ハードウェアとソフトウェアの切り分けの最適化を行うことを特徴とする請求項4に記載のアルゴリズム記述におけるバスの性能評価方法。

【請求項7】 アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、

前記バスのトラフィックをカウントする評価関数を作成する工程と、

前記ソースを構文解析しながら行単位で順次読み込む工程と、

前記ソースの記述が評価対象となるバスに乗るあらかじめ定義された変数へのデータの書き込みを表しているか否かを判定する工程と、

前記判定の結果、データの書き込みを表していた場合、前記作成した評価関数をその直前もしくは直後に埋め込み前記ソースの記述を修正する工程と、

前記読み込んだソース構文が最後の行になるまで前記各工程の処理を繰り返し、コンパイルしてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築し、これを実行して評価対象となるバスのトラフィックを算出する工程と、

既知の処理レートに対するバストラフィックを算出して評価対象となるバスの性能評価を行う工程とを備えたことを特徴とするアルゴリズム記述におけるバスの性能評価方法。

【請求項8】 前記評価対象となるバスに乗るそれぞれの変数のビット幅を

n、前記評価対象のバスを構成するビット幅をm（但し、 $n \geq m$ ）とした場合、前記処理レートに対するバストラフィックは、データ転送回数に n/m 倍したものを処理レートで除算することによって算出して性能評価を行うことを特徴とする請求項7に記載のアルゴリズム記述におけるバスの性能評価方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、アーキテクチャ設計を上位設計の段階で行うために、アルゴリズム検証用に、汎用高級言語（CやC⁺⁺）で記述したソースを用いて、ハードウェア（以下、単にH/Wという）、ソフトウェア（以下、単にS/Wという）をつなぐバスの性能評価を行う、アルゴリズム記述におけるバスの性能評価方法に関する。

【0002】

【従来の技術】

半導体技術の進歩に伴い、ボードに多数のLSIチップを配置してシステムを具現化する形態から、単体のLSIチップでシステムを具現化する形態が増えてきた。そのため、従前のLSIでは、外部端子として接続していた信号が内部信号となり、LSIに内蔵されている。従って、従来のようにボードを使用したシステム検証を行うためには、内部信号をすべて外部端子とした評価用のLSIを別途製造しなければならない、実質以前のようなボードを使用した検証を行うことが困難になってきた。

【0003】

図11に従来のLSIの設計開発の流れをフローチャートで示す。ロジック回路、システム及びデバイスを製造する際、通常は、まず、ハードウェア、ソフトウェアを意識しないシミュレーションプラットフォームを構築し（ステップB1）、アルゴリズムが正しいか否かを検証する（ステップB2）。次に、構築したシミュレーションプラットフォームから、ハードウェア、ソフトウェアに切り分ける。このハードウェア、ソフトウェアへの切り分けは、経験的に行われてきた。

【0004】

H/W設計において、まずアルゴリズムと等価な機能を持つソースを一般的にはHDL (Hardware Description Language) 言語で記述し、回路合成を行う(ステップB3)。次に、作成したソースが正常に動作するか否かを検証する(ステップB4)。S/W設計において、まずアルゴリズムと等価な機能を持つソースを一般的には、各CPUに依存したプログラミング言語で記述する(ステップB5)。次に、作成したソースが正常に動作するか否かを検証する(ステップB6)。最後に、ハードウェア、ソフトウェアを結合した協調検証が行われる(ステップB7)。

【0005】

【発明が解決しようとする課題】

上述したように、LSIを製造するにあたり、実際に作り込む前にシステムシミュレーションを行い、バスの性能評価を行い、ハードウェア、ソフトウェアの切り分けを行うアーキテクチャ設計をシミュレーションにより行うことが必須条件となってきた。従来、システムシミュレーションは上述したようにハードウェア、ソフトウェアを統合した協調検証により行われてきた。

【0006】

近年、製造する回路規模の増加と共に、作成する回路のコード数が多大なものとなってきた。従って、HDL言語、各CPUに依存するプログラミング語で記述された協調検証のシミュレーションスピードが非常に遅くなり、実質、協調検証によるアーキテクチャ設計を行うことが困難になってきた。

また、協調検証によりアーキテクチャ設計を行った場合、ハードウェア、ソフトウェアをつなぐバスの性能評価結果により、ハードウェア、ソフトウェアをつなぐバスの変更等が生じた場合、HDL言語、各CPUに依存したプログラミング言語を修正する必要がある(図11、ステップB8、B9)。

【0007】

この場合、作成する回路のコード数の増加に従い、作成する回路記述がより複雑なものになってきたため回路記述の修正が複雑なものとなり、協調検証からのフィードバックに要する時間と費用が多大なものとなっている。従って、回路規模が増大し、開発期間が短縮されてきている近年において、ハードウェア、ソフ

トウェアをつなぐバスの性能評価を上位設計で行い、アーキテクチャー設計を上位設計の段階で行うことが設計開発において必須条件となってきた。

【0008】

本発明は上記事情に鑑みてなされたものであり、アルゴリズム設計で作成した汎用高級言語で記述されたソースをハードウェア、ソフトウェアに切り分けた後、バスのトラフィックをカウントする評価関数を作成して、バスにデータ(変数)が乗る毎に、その評価関数を実行するように、ソースを修正し、その後、性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減を可能としたアルゴリズム記述におけるバスの性能評価方法を提供することを目的とする。

【0009】

【課題を解決するための手段】

上記した課題を解決するために請求項1に記載の発明は、アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、ソースを、バスにデータ転送が生じた際に、特定の評価関数を実行するように修正することにより、アーキテクチャー設計を行うためのシミュレーションプラットフォームを構築することとした。

このことにより、アルゴリズム設計で用いた、例えば、C言語、C++言語等汎用高級言語で記述されたソースを、評価対象となるバスに乗る変数にデータの書き込みが生じた際に、一定の値をインクリメントする特定の評価関数を実行するように修正し、シミュレーションプラットフォームを構築するため、設計開発フローの中で、上位設計の段階において、ハードウェア、ソフトウェアをつなぐバスの処理レートに対するバストラフィックを算出し、バスの性能評価を行うことができる。

【0010】

請求項2に記載の発明は、請求項1に記載のアルゴリズム記述における性能評価方法において、シミュレーションプラットフォームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、バスの性能評価を上位設計の段階において行うこととした。

このことにより、構築されたシミュレーションプラットフォームを用いて性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減が可能となる。

【0011】

請求項3に記載の発明は、請求項1または2に記載のアルゴリズム記述における性能評価方法において、評価対象となるバスの性能評価結果を前記アルゴリズム検証で用いたソースにフィードバックすることにより、上位設計の段階において、アーキテクチャー設計を行うこととした。

このことにより、上位設計の段階で、評価対象となるバスの性能評価結果をフィードバックするため、設計開発フローの上位設計段階において、アーキテクチャー設計を行うことができる。

【0012】

請求項4に記載の発明は、アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、ソースを用いてアーキテクチャー設計を行うためのシミュレーションプラットフォームを構築し、構築したシミュレーションプラットフォームを用いて評価対象となるバスの性能評価を行い、その評価結果に基づきソースを修正してシミュレーションプラットフォームを再構築し、再構築されたシミュレーションプラットフォームを用いて性能評価を行うこととした。

このことにより、アルゴリズム設計で用いた、例えば、C言語、C++言語等汎用高級言語で記述されたソースを、評価対象となるバスに乗る変数にデータの書

き込みが生じた際に、一定の値をインクリメントする特定の評価関数を実行するように修正し、シミュレーションプラットフォームを構築するため、設計開発フローの中で、上位設計の段階において、ハードウェア、ソフトウェアをつなぐバスの処理レートに対するバストラフィックを算出し、バスの性能評価を行うことができる。

【 0 0 1 3 】

請求項 5 に記載の発明は、請求項 4 に記載のアルゴリズム記述におけるバスの性能評価方法において、ソースを用いてアーキテクチャ設計を行うためのシミュレーションプラットフォームを構築するために、ソースを所定の単位で解析することによりハードウェアとソフトウェア記述に分け、バスのトラフィックをカウントする評価関数を作成し、バスを使用したデータ転送が発生する都度、前記作成された評価関数を実行するように前記ソース構文を修正することとした。

このことにより、上位設計の段階で、評価対象となるバスの性能評価結果をフィードバックするため、設計開発フローの上位設計段階において、アーキテクチャ設計を行うことができる。また、アルゴリズム設計で用いた C 言語、C++ 言語等汎用高級言語で記述されたソースによりアーキテクチャ設計を行うため、アーキテクチャ設計に要するシミュレーション時間を大幅に短縮することができる。

【 0 0 1 4 】

請求項 6 に記載の発明は、請求項 4 に記載のアルゴリズム記述におけるバスの性能評価方法において、性能評価を行うために、評価関数に基づいて性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、ハードウェアとソフトウェアの切り分けの最適化を行うこととした。

このことにより、例えば、C 言語、C++ 言語等汎用高級言語で記述されたソースによりアーキテクチャ設計を行うため、アーキテクチャ設計によるフィードバックを容易化でき、アーキテクチャ設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証における切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減を可能とする

【0015】

請求項7に記載の発明は、アルゴリズム検証用に汎用高級言語で記述されたソースを用いてハードウェアとソフトウェアをつなぐバスの性能評価を行うアルゴリズム記述におけるバスの性能評価方法であって、バスのトラフィックをカウントする評価関数を作成する工程と、ソースを構文解析しながら行単位で順次読み込む工程と、ソースの記述が評価対象となるバスに乗るあらかじめ定義された変数へのデータの書き込みを表しているか否かを判定する工程と、判定の結果、データの書き込みを表していた場合、作成した評価関数をその直前もしくは直後に埋め込み前記ソースの記述を修正する工程と、読み込んだソース構文が最後の行になるまで前記各工程の処理を繰り返し、コンパイルしてアーキテクチャ設計を行うためのシミュレーションプラットフォームを構築し、これを実行して評価対象となるバスのトラフィックを算出する工程と、既知の処理レートに対するバストラフィックを算出して評価対象となるバスの性能評価を行う工程とを備えることとした。

このことにより、アルゴリズム設計で作成した汎用高級言語で記述されたソースをハードウェア、ソフトウェアに切り分けた後、バスのトラフィックをカウントする評価関数を作成して、バスにデータ(変数)が乗る毎に、その評価関数を実行するように、ソースを修正し、その後、性能評価を行い、最終的に算出された処理レートに対するバストラフィックにより、アーキテクチャ設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減が可能となる。

【0016】

請求項8に記載の発明は、請求項7に記載のアルゴリズム記述におけるバスの性能評価方法において、評価対象となるバスに乗るそれぞれの変数のビット幅を n 、評価対象のバスを構成するビット幅を m （但し、 $n \geq m$ ）とした場合、処理レートに対するバストラフィックは、データ転送回数に n/m 倍したものを処理レートで除算することによって算出して性能評価を行うこととした。

このことにより、評価対象となるバスに乗る変数のビット幅が異なり、且つ評価対象となるバスのビット幅より大きい場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャー設計を行うことが可能となる。

【0017】

【発明の実施の形態】

図1は、本発明に従うLSIの設計開発の流れをフローチャートで示した図である。図1において、アーキテクチャー設計を上位設計の段階において行うために、従来から使用されていた設計開発フロー（図11）に、シミュレーションプラットフォームを構築する工程と、性能評価工程（ステップA15）が追加されている。以下にその詳細を記述する。

【0018】

まずアルゴリズム設計を行い（ステップA1）、製造するロジック回路、デバイス及びシステムのアルゴリズムが正しいか否かをアルゴリズム検証により検証する（ステップA2）。

次に、アルゴリズム設計において使用されたソースを用いて、アーキテクチャー設計を行うためのシミュレーションプラットフォームを構築する。ここでは、まず、ハードウェア、ソフトウェアを切り分ける（ステップA3）。次に、評価関数を作成する（ステップA4）。ここで、評価関数は、一定量をカウントする関数であれば良い。

【0019】

次に、ハードウェア、ソフトウェアをつなぐバスに乗る変数を選択する（ステップA5）。そして、アルゴリズム設計で用いたソースを、評価対象となるバスに乗る変数にデータの書き込みが生じた際、すなわち評価対象となるバスにデータ転送が生じた際に、作成した評価関数を実行するように修正し、アーキテクチャー設計を行うためのシミュレーションプラットフォームを構築する（ステップA6）。

【0020】

次に、構築したシミュレーションプラットフォームを用いて、評価対象となるバスの性能評価を行う。まず、構築したシミュレーションプラットフォームを用いて、検証を行う(ステップA7)。そして、作成した評価関数により方法は異なるが、バストラフィックを算出する(ステップA8)。ここでは、メイン関数に要求される処理レートが既知であることから、評価対象となるバスの処理レートに対するバストラフィックを算出する(ステップA9)。

算出した処理レートに対するバストラフィックにより、ハードウェア、ソフトウェアの切り分け及びバス構成の妥当性の検討ができる。検討後、バスの変更等が必要な場合、汎用高級言語、例えば、C言語やC++言語で記述されたソースを修正して、再度、シミュレーションプラットフォームの構築、性能評価を行う。このように評価対象となるバスの性能評価結果をフィードバックすることにより(ステップA16)、上位設計の段階におけるアーキテクチャ設計を実現する。

【0021】

アーキテクチャ設計を行った後、ハードウェア、ソフトウェアそれぞれの設計フローに入る。H/W設計において、まずアルゴリズムと等価な機能を持つソースを一般的にはHDL言語で記述し、回路合成を行う(ステップA10)。次に、作成したソースが正常に動作するか否かを検証する(ステップA11)。S/W設計において、まずアルゴリズムと等価な機能を持つソースを一般的には、各CPUに依存したプログラミング言語で記述する(ステップA12)。そして、作成したソースが正常に動作するか否かを検証する(ステップA13)。

【0022】

最後に、ハードウェア、ソフトウェアを結合した協調検証を行う(ステップA14)。アーキテクチャ設計は、本発明の設計開発フローにおいては、既に上位設計の段階において終了しているため、協調検証は、動作確認を行うのみのシミュレーションと位置づけ、設計を行わない。従って、協調検証からのフィードバックを行う必要がなくなる。

【0023】

図2に本発明の一実施形態をリスト形式で示す。図2を参照すれば、評価対象となるバスに乗る変数として、a、b、cの3つの変数を持ち、いずれもグロー

バル定義されている。また、評価関数 $BUS0()$ と、ローカル変数 b_{us0} を持つ。ここでは、評価関数は、スタティック変数 i をインクリメントし、戻り値とする関数である。ローカル変数 b_{us0} は、評価関数 $BUS0()$ からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。また、評価対象となるバスに乗る変数 a 、 b 、 c のビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。

【0024】

図2に示す実施形態では、評価対象となるバスに乗る変数 a 、 b 、 c にデータの書き込みが行われるとき、必ず評価関数 $BUS0()$ を実行するように、直後に評価関数 $BUS0()$ が埋め込まれている。評価関数 $BUS0()$ は、実行される度に、スタティック変数 i を1だけインクリメントし、戻り値とするため、評価関数 $BUS0()$ からの戻り値は、評価対象となるバスに乗る変数のデータ書き込み回数を表すことになる。

【0025】

評価対象となるバスに乗る変数のデータ書き込み回数は、評価対象となるバスのデータ転送、すなわちバストラフィックを表している。メイン関数に要求される処理レートが既知であるため、下記式1により処理レートに対するバストラフィックを算出し、性能評価を行うことができる。

(処理レートに対するバストラフィック)

$$= (\text{データ転送回数}) / (\text{処理レート}) \quad \dots \quad (1)$$

算出した処理レートに対するバストラフィックにより、ハードウェア、ソフトウェアをつなぐバスを変更する場合、アルゴリズム設計で用いられたソースを修正し、再度シミュレーションプラットフォームを構築する。

【0026】

図3に再度構築したシミュレーションプラットフォームの一例を示す。図3を参照すれば、再度構築したシミュレーションプラットフォームの一例は、図2に示す実施形態における評価対象となるバスの性能評価結果により、変数 b を評価対象となるバスに乗らない変数としている。従って、評価対象となるバスに乗る変数として、 a 、 c の2つの変数のみを持つ。変数 b はバスに乗らないので、変数 b

にデータの書き込みが行われる直後には、評価関数 BUS0 () が埋め込まれておらず、変数 a、c にデータの書き込みが行われるときには、直後に評価関数 BUS0 () が埋め込まれ、評価関数 BUS0 () を必ず実行するようになっている。

シミュレーションプラットフォームを再度構築した後、シミュレーションにより、検証を行う。再度、式 1 により処理レートに対するバストラフィックを算出し、評価対象となるバスの性能評価を行う。

【 0 0 2 7 】

以上説明のように、評価対象となるバスのデータレートに対する平均バストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【 0 0 2 8 】

図 4 に、図 1 に示す実施形態の動作がフローチャートで示されている。

図 4 を参照すると、まず特定の評価関数を作成する(ステップ C 1)。評価関数は、ある一定の値をインクリメントする関数であれば何でも良い。次に、アルゴリズム設計で用いた C 言語、もしくは C++ 言語で記述されたソースを、構文解析を行いながら 1 行ずつ読み込む(ステップ C 2)。

【 0 0 2 9 】

次に、読み込んだソースの記述が、評価対象となるバスに乗る変数へのデータの書き込みを表しているか否かを判定する(ステップ C 3)。評価対象となるバスに乗る変数へのデータの書き込みを表している場合、その直後に作成した評価関数を埋め込む(ステップ C 4)。この作業により、評価対象となるバスに乗る変数にデータの書き込みが行われた場合、すなわち評価対象となるバスにデータ転送が生じた場合に、必ず作成した評価関数を実行するようになる。

【 0 0 3 0 】

以上の作業を読み込んだソースの記述が、アルゴリズム設計で用いたソースの最後の行まで行い、アルゴリズム設計で用いたソースを修正する(ステップ C 5)。アルゴリズム設計で用いられたソースの最後の行まで修正が行われた後、コンパイルをして、アーキテクチャ設計を行うためのシミュレーションプラットフォーム

ームを構築する(ステップC6)。次に、それを実行し、評価対象となるバスのバストラフィックを算出する(ステップC7)。そして、メイン関数に要求される処理レートが既知であることから、処理レートに対するバストラフィックを算出し、評価対象となるバスの性能評価を行う(ステップC8)。

【0031】

図5乃至図10は本発明の他の実施形態をリスト形式で示した図である。以下、本発明の他の実施形態につき、図2に示す実施形態と対比しながら詳細に説明する。

【0032】

図5に示す実施形態は、評価対象となるバスに乗る変数として、a、b、cの3つの変数を持ち、いずれもローカル定義されている。また、評価関数BUS0()と、ローカル変数bus0()を持つ。ここでは、評価関数は、スタティック変数iをインクリメントし、戻り値とする関数である。ローカル変数bus0()は、評価関数BUS0()からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【0033】

ここでは、評価対象となるバスに乗る変数a、b、cのビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとし、評価対象となるバスに乗る変数が、グローバル定義ではなく、ローカル定義されていることが図2に示す実施形態と異なる。ここで性能評価を行う場合、上記式1により処理レートに対するバストラフィックを算出して行う。

【0034】

以上説明のように、評価対象となるバスに乗る変数がグローバル定義ではなく、ローカル定義されている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【0035】

図6に示す実施形態は、評価対象となるバスに乗る変数として、a、b、cの3つの変数を持ち、いずれもグローバル定義されている。また、評価関数BUS

0 () と、ローカル変数 bus 0 () を持つ。ここでは、評価関数は、スタティック変数 i をインクリメントし、戻り値とする関数である。ローカル変数 bus 0 () は、評価関数 BUS 0 () からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【 0 0 3 6 】

評価対象となるバスに乗る変数 a、b、c のビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。ここで、評価対象となるバスに乗る変数にデータの書き込みが生じた際、すなわち評価対象となるバスにデータ転送が生じた際に、必ず評価関数を実行するように直前に評価関数 BUS 0 () が埋め込まれている点が図 2 に示す実施形態とは異なる。ここで性能評価を行う場合、式 1 により処理レートに対するバストラフィックを算出して行う。

【 0 0 3 7 】

以上説明のように、評価対象となるバスに乗る変数にデータの書き込みが生じた際、すなわち評価対象となるバスにデータ転送が生じた際に、必ず評価関数を実行するように直前に評価関数 BUS 0 () が埋め込まれている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【 0 0 3 8 】

なお、ここで、評価対象となるバスに乗る変数 a、b、c のビット幅がすべて同一で、評価対象となるバスのビット幅より大きい場合、例えば、評価対象となるバスに乗る変数 a、b、c のビット幅が n ビット、評価対象となるバスのビット幅が m ビット (但し、 $n \geq m$) の場合について説明する。具体的に、 $n = 32$ 、 $m = 8$ を想定する。

この例において、評価対象となるバスに乗る変数にデータの書き込みが行われた場合、評価対象となるバスにデータ転送は、 n / m 回、すなわち 4 回行われる。従って、性能評価を行う場合、下記式 2 により処理レートに対するバストラフィックを算出して行う。

(処理レートに対するバストラフィック)

$$= (\text{実行回数}) \times 4 / (\text{処理レート}) \quad \dots \quad (2)$$

【 0 0 3 9 】

次に、図 7 以降を参照して本発明の更に他の実施形態について説明する。

図 7 は、評価対象となるバスに乗る変数として、 a 、 b の 2 つの変数を持ち、いずれもグローバル定義されている。また、評価関数 $BUS1()$ と、ローカル変数 b_{us1} を持つ。ここで、評価関数は、引数として bit と b_{us} を持ち、スタティック変数 i を (bit/b_{us}) 分インクリメントし、戻り値とする関数である。ローカル変数 b_{us1} は、評価関数 $BUS1()$ からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【 0 0 4 0 】

ここでは、評価対象となるバスに乗る変数 a のビット幅は 32 ビット、評価対象となるバスに乗る変数 b のビット幅は 16 ビットとする。また、評価対象となるバスのビット幅は 8 ビットとする。

評価対象となるバスに乗る変数のビット幅が異なり、評価対象となるバスのビット幅より大きい点で図 2 に示す実施形態とは異なる。また、評価関数が引数として bit と b_{us} を持ち、スタティック変数 i を (bit/b_{us}) 分インクリメントし、戻り値とする関数であるという点も図 2 に示す実施形態と異なる。ここで性能評価を行う場合、上述した式 1 により処理レートに対するバストラフィックを算出して行う。

【 0 0 4 1 】

以上説明のように、評価対象となるバスに乗る変数のビット幅が異なり、且つ評価対象となるバスのビット幅より大きい場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【 0 0 4 2 】

図 8 に示す実施形態は、評価対象となるバスに乗る変数として、 a 、 b の 2 つの変数を持ち、いずれも配列で定義されている。また、評価関数 $BUS2()$ と、ローカル変数 b_{us2} を持つ。更に、評価関数は、引数として ele を持ち、

スタティック変数 i を $e1e$ 分インクリメントし、戻り値とする関数である。ローカル変数 $b u s 2$ は、評価関数 $B U S 2 ()$ からの戻り値を格納する変数であり、評価対象となるバスのデータ転送回数を表す。

【0043】

ここでは、評価対象となるバスに乗る変数 a 、 b のビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。また、評価対象となるバスに乗る変数は、配列で定義されており、メイン関数内においてアドレス転送が行われている点が図2に示す実施形態と異なる。更に、評価関数が引数として $e1e$ を持ち、スタティック変数 i を $e1e$ 分インクリメントし、戻り値とする関数であるという点も図2に示す実施形態と異なる。

ここで性能評価を行う場合、上述した（式1）により処理レートに対するバストラフィックを算出して行う。

【0044】

以上説明のように、評価対象となるバスに乗る変数が配列で定義され、メイン関数内でアドレス転送が行われている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【0045】

図9に示す実施形態は、評価対象となるバスに乗る変数として、 a 、 b の2つの変数を持ち、いずれもグローバル定義されている。また、評価関数 $B U S 3 ()$ と、グローバル変数 $b u s$ を持つ。評価関数は、グローバル変数 $b u s$ をインクリメントする関数である。グローバル変数 $b u s$ は、評価対象となるバスのデータ転送回数を表す。

ここでは、評価対象となるバスに乗る変数 a 、 b のビット幅は、すべて同一とし、評価対象となるバスのビット幅より小さいものとする。評価関数内でインクリメントする変数がグローバル定義されている点が図2に示す実施形態と異なる。

ここで性能評価を行う場合、滋養術した式1により処理レートに対するバスト

ラフィックを算出して行う。

【0046】

以上説明のように、評価関数内でインクリメントする変数がグローバル定義されている場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【0047】

図10に、シミュレーションプラットフォームをC++言語で記述したソースにより構築した例が示されている。

図10に示す実施形態は、BUSクラス定義された評価対象となるバスbus0、bus1、bus2と、グローバル定義された評価対象となるバスbus0に乘る変数aと、ローカル定義された評価対象となるバスbus1に乘る変数bと、配列で定義された評価対象となるバスbus2に乘る変数cを持つ。

【0048】

また、ここでは、BUSクラスを持つ。BUSクラス内には、メンバ変数iを1インクリメントする関数count()と、引数bit、busを持つ。更に、メンバ変数iを(bit/bus)分インクリメントする関数Count_bit()と、引数eleを持ち、メンバ変数iをele分インクリメントする関数count_hairetu()が定義されている。従って、BUSクラス内のメンバ変数iは、評価対象となるバスのデータ転送回数を表す。

【0049】

評価対象となるバスに乘る変数a、cのビット幅は、すべて同一とし、評価対象となるバスそれぞれbus0、bus1のビット幅より小さいものとする。また、評価対象となるバスに乘る変数bのビット幅は32ビット、評価対象となるバスbus1のビット幅は8ビットとする。

ここでは、ソースがC++言語で記述されている点で上述した図2を含む各実施形態と異なり、性能評価を行う場合、(式1))により処理レートに対するバストラフィックを算出して行う。

【0050】

以上説明のように、シミュレーションプラットフォームをC++言語で記述したソースにより構築した場合においても、処理レートに対するバストラフィックを算出し、性能評価を行い、その性能評価結果をフィードバックすることにより、上位設計の段階でのアーキテクチャ設計を行うことが可能となる。

【0051】

以上説明のように本発明は、アルゴリズム設計で使用されるC言語、C++言語等汎用高級言語で記述されたソースを、評価対象となるハードウェア、ソフトウェアをつなぐバスにデータ転送が生じた際に、特定の評価関数を実行するように修正することでアヘクアーキテクチャ設計を行うためのシミュレーションプラットフォームを構築し、ここで構築されたシミュレーションプラットフォームを用いて、シミュレーションを行うことにより、処理レートに対するバストラフィックを算出し、バスの性能評価を上位設計の段階において行うものである。

【0052】

また、評価対象となるバスの性能評価結果をアルゴリズム検証で用いたソースにフィードバックすることにより、上位設計の段階において、アーキテクチャ設計を行うことができ、アーキテクチャ設計に要するシミュレーション時間を短縮することができる。更に、ハードウェア、ソフトウェアを結合した協調検証を動作確認のみを行うシミュレーションと位置づけ、実際の設計を行わないことにより、ロジック回路、システム及びデバイスの設計開発全体に要する時間と費用を大幅に削減することができる。

【0053】

【発明の効果】

以上説明のように本発明によれば、アルゴリズム設計で用いた、例えば、C言語、C++言語等汎用高級言語で記述されたソースを、評価対象となるバスに乗る変数にデータの書き込みが生じた際に、一定の値をインクリメントする特定の評価関数を実行するように修正することによりシミュレーションプラットフォームを構築するため、設計開発フローの中で、上位設計の段階において、ハードウェア、ソフトウェアをつなぐバスの処理レートに対するバストラフィックを算出し、バスの性能評価を行うことができる。

また、上位設計の段階で、評価対象となるバスの性能評価結果をフィードバックするため、設計開発フローの上位設計段階において、アーキテクチャー設計を行うことができる。

更に、アルゴリズム設計で用いたC言語、C++言語等汎用高級言語で記述されたソースによりアーキテクチャー設計を行うため、アーキテクチャー設計に要するシミュレーション時間を大幅に短縮することができる。一般的にC言語、C++言語は、HDL言語に比較してシミュレーションスピードが約1000倍速い。

【0054】

また本発明によれば、C言語、C++言語等汎用高級言語で記述されたソースによりアーキテクチャー設計を行うため、アーキテクチャー設計によるフィードバックを容易化でき、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行うことで、実際のコーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排することができ、設計ターンアラウンドタイムの大幅な削減を可能とする。更に、C言語、C++言語等汎用高級言語は、HDL言語、アセンブラ言語と比較して少ないコード数での記述が可能であり、変更、修正が容易にできる。

【0055】

また、ハードウェア、ソフトウェアを結合した協調検証を動作確認のみを行うシミュレーションと位置づけ、設計を行わないため、協調検証からのフィードバックを無くし、RTLでのシミュレーション回数を削減できるため回路設計全体に要する時間と費用を大幅に削減できる。このことは、製造する回路が大きければ大きいほど、その効果は顕著に現れる。

【図面の簡単な説明】

【図1】 本発明に従うLSIの設計開発の流れをフローチャートで示した図である。

【図2】 本発明の一実施形態をリスト形式で示した図である。

【図3】 図2に示す本発明実施形態において再構築したシミュレーションプラットフォームの一例を示す図である。

【図4】 図2に示す実施形態における作業手順をフローチャートで示した

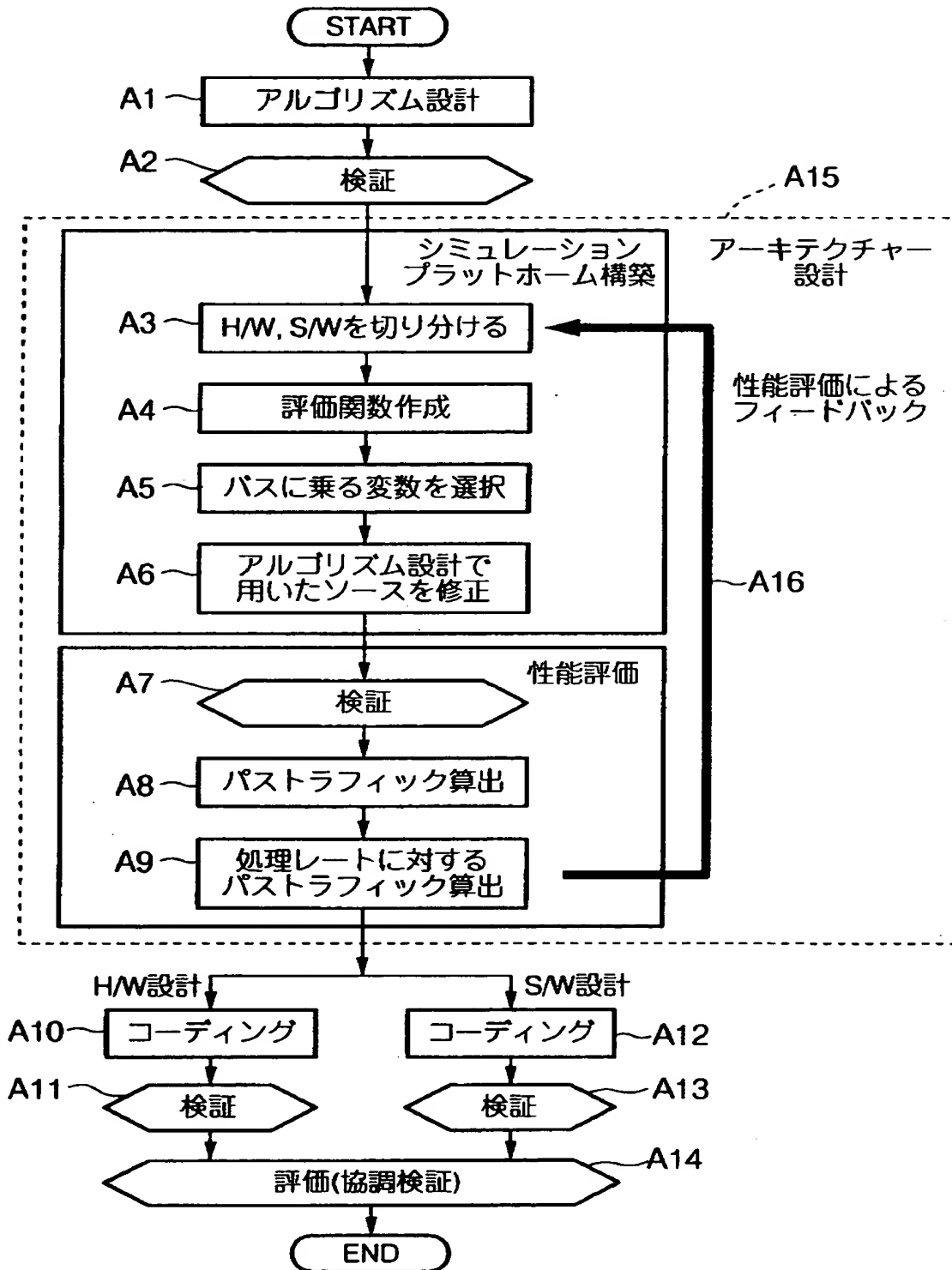
図である。

- 【図 5】 本発明の他の実施形態をリスト形式で示した図である。
- 【図 6】 本発明の更に他の実施形態をリスト形式で示した図である。
- 【図 7】 本発明の更に他の実施形態をリスト形式で示した図である。
- 【図 8】 本発明の更に他の実施形態をリスト形式で示した図である。
- 【図 9】 本発明の更に他の実施形態をリスト形式で示した図である。
- 【図 1 0】 本発明の更に他の実施形態をリスト形式で示した図である。
- 【図 1 1】 従来の L S I の設計開発の流れをフローチャートで示した図で

ある。

【書類名】 図面

【図1】



【図 2】

```
#include <stdio.h>

int BUS0(void);

int a;          /* 評価対象となるバスに乗る変数 */
int b;          /* 評価対象となるバスに乗る変数 */
int c;          /* 評価対象となるバスに乗る変数 */

/* メイン関数 */
void main(void)
{
    int bus0;
    .
    .
    .
    a=0;          /* バスに乗る変数に書き込みが発生した場合 */
    bus0=BUS0();  /* 直後に評価関数を埋め込む */
    .
    .
    .
    b=1;
    bus0=BUS0();
    .
    .
    .
    c=10;
    bus0=BUS0();
    .
    .
    .
    a=3;
    bus0=BUS0();
    b=4;
    bus0=BUS0();

    printf("BUS Traffic=%d[回]*n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
    static int i=0;
    i++;          /* スタティック変数iをインクリメント */
    return i;
}
```

【図 3】

```

#include <stdio.h>

int BUS0(void);

int a;          /* 評価対象となるバスに乗る変数 */
int b;          /* 評価対象となるバスに乗る変数 */
int c;

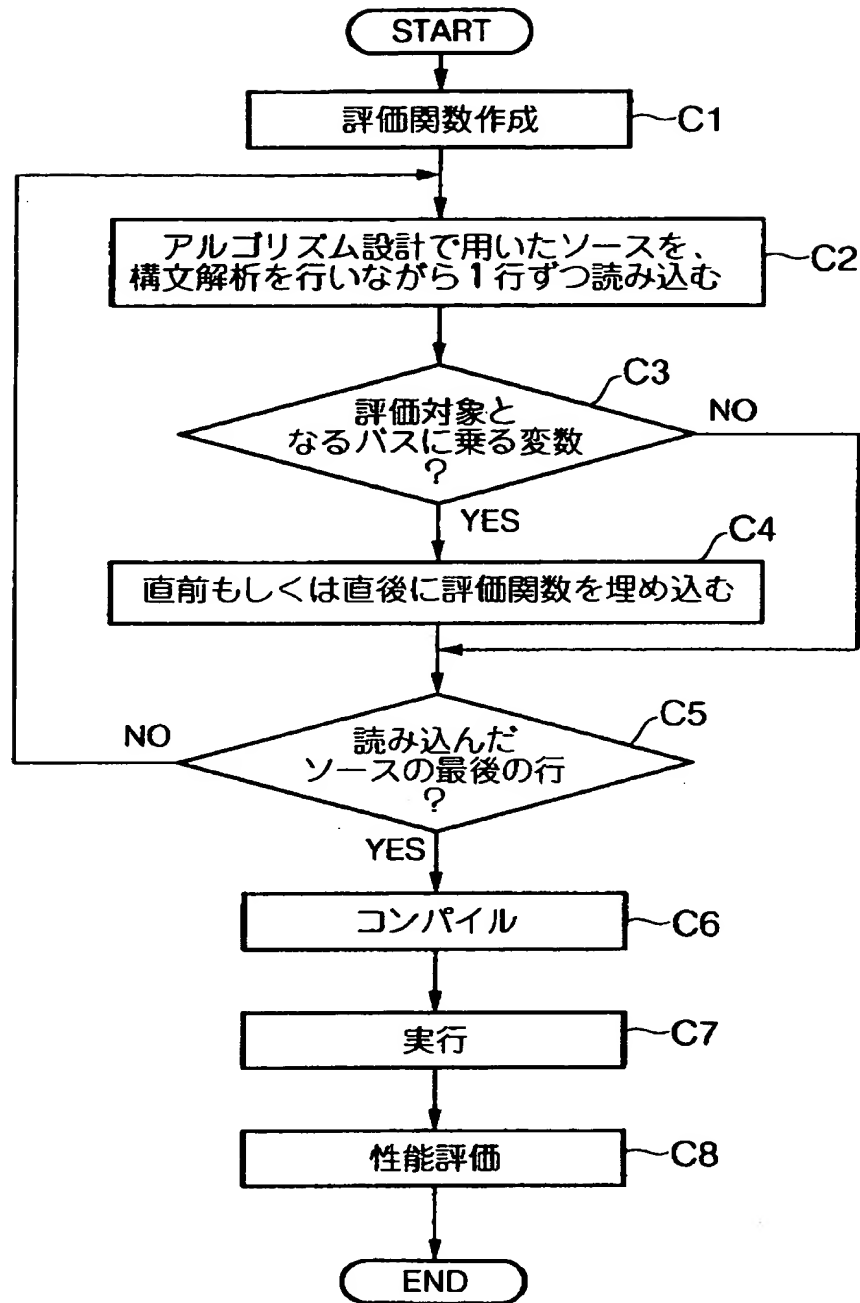
/* メイン関数 */
void main(void)
{
    int bus0;
    .
    .
    a=0;          /* バスに乗る変数に書き込みが発生した場合 */
    bus0=BUS0();  /* 直後に評価関数を埋め込む */
    .
    .
    b=1;          /* 変数bは、バスに乗らないので、評価関数は埋め込まない */
    .
    .
    c=10;
    bus0=BUS0();
    .
    .
    a=3;
    bus0=BUS0();
    b=4;          /* 変数bは、バスに乗らないので、評価関数は埋め込まない */

    printf("BUS Traffic=%d[回]*n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
    static int i=0;
    i++;          /* スタティック変数iをインクリメント */
    return i;
}

```

【図 4】



【図 5】

```
#include <stdio.h>

int BUS0(void);

/* メイン関数 */
void main(void)
{
    int a;                /* 評価対象となるバスに乗る変数 */
    int b;                /* 評価対象となるバスに乗る変数 */
    int c;                /* 評価対象となるバスに乗る変数 */

    int bus0;
    .
    .
    .
    a=0;                  /* バスに乗る変数に書き込みが発生した場合 */
    bus0=BUS0();          /* 直後に評価関数を埋め込む */
    .
    .
    .
    b=1;
    bus0=BUS0();
    .
    .
    .
    c=10;
    bus0=BUS0();
    .
    .
    .
    a=3;
    bus0=BUS0();
    b=4;
    bus0=BUS0();

    printf("BUS Traffic=%d[回]*n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
    static int i=0;       /* スタティック変数iをインクリメント */
    i++;
    return i;
}
```

【図 6】

```

#include <stdio.h>

int BUS0(void);

int a;          /* 評価対象となるバスに乗る変数 */
int b;          /* 評価対象となるバスに乗る変数 */
int c;          /* 評価対象となるバスに乗る変数 */

/* メイン関数 */
void main(void)
{
    int bus0;    /* バスに乗る変数に書き込みが発生した場合 */
                /* 直前に評価関数を埋め込む */

    bus0=BUS0();
    a=0;

    bus0=BUS0();
    b=1;

    bus0=BUS0();
    c=10;

    bus0=BUS ();
    a=3;
    bus0=BUS0();
    b=4;

    printf("BUS Traffic=%d[回]¥n", bus0);
}

/* 評価関数 */
int BUS0(void)
{
    static int i=0;
    i++;          /* スタティック変数iをインクリメント */
    return i;
}

```

【図 7】

```
#include <stdio.h>

int BUS1(int, int);

int a;          /* 評価対象となるバスに乗る変数(32ビット) */
int b;          /* 評価対象となるバスに乗る変数(16ビット) */

/* メイン関数 */
void main(void)
{
    int bus1;
    .
    .
    .
    a=7;          /* バスに乗る変数に書き込みが発生した場合 */
    bus1=BUS1(32,8); /* 直後に評価関数を埋め込む */
    .
    .
    .
    bus1=BUS1(32,8);
    a=6;
    bus1=BUS1(16,8);
    b=10;
    printf("BUS Traffic=%d[回]*n", bus1);
}

/* 評価関数 */
int BUS1(int bit, int bus) /* バスのデータ転送分インクリメントする関数 */
{
    static int i=0;
    i+=bit / bus;
    return i;
}
```

【図 8】

```
#include <stdio.h>

int BUS2(int);

int a[10];          /* 評価対象となるバスに乗る変数(配列) */
int b[20];          /* 評価対象となるバスに乗る変数(配列) */

/* メイン関数 */
void main(void)
{
    int bus2;
    int *adr;
    .
    .
    .
    adr=&a[0];        /* バスに乗る変数のアドレス転送が発生した場合 */
    bus2=BUS2(10);    /* 直後に評価関数を埋め込む */
    .
    .
    .
    adr=&a[0];
    bus2=BUS2(10);
    adr=&b[0];
    bus2=BUS2(20);
    printf("BUS Traffic=%d[回]¥n", bus2);
}

/* 評価関数 */
int BUS2(int ele)    /* 要素数分インクリメントする関数 */
{
    static int i=0;
    i+=ele;
    return i;
}
```


【図 9】

```
#include <stdio.h>

void BUS3(void);

int a;          /* 評価対象となるバスに乗る変数 */
int b;          /* 評価対象となるバスに乗る変数 */

int bus=0;      /* 評価関数内でインクリメントされる変数 */

/* メイン関数 */
void main(void)
{
    .
    .
    .
    a=0;          /* バスに乗る変数に書き込みが発生した場合 */
    BUS3();       /* 直後に評価関数を埋め込む */
    .
    .
    .
    a=1;
    BUS3();
    b=10;
    BUS3();
    printf("BUS Traffic=%d[回]¥n", bus);
}

/* 評価関数 */
void BUS3(void)
{
    bus++;        /* グローバル変数busをインクリメント */
}
```

【図 1 0】

```

#include <stdio.h>

/* BUSクラス */
class BUS
{
    int i;

public:
    BUS():i(0){}

    int count(){return i + 1;}          /* iをインクリメントする関数 */
    int count_bit(int bit, int bus)    /* バスのデータ転送分インクリメントする関数 */
    {
        i += bit/bus;
        return i;
    }
    int count_hairetu(int, ele)        /* 要素分インクリメントする関数 */
    {
        i += ele;
        return i;
    }

    void print () {printf("BUS Traffic=%d[回]\n",i);}
};

int a;                                /* bus0に乗る変数 */

/* メイン関数 */
int main()
{
    BUS bus0;                          /* 評価対象となるバス(32ビット) */
    BUS bus1;                          /* 評価対象となるバス(8ビット) */
    BUS bus2;                          /* 評価対象となるバス(32ビット) */

    int b;                             /* bus1に乗る変数(32ビット) */
    int c[10];                         /* bus2に乗る変数(配列) */

    int *cp;

    int i;

    a=0;                               /* 評価対象となるバスに乗る変数にデータ書き込みが */
    bus0.count();                      /* 行われた場合、直後に評価関数を埋め込む */

    b=1;
    bus1.count_bit(32, 8);

    cp=&c[0];
    bus2.count_hairetu(10);

    for(i=0; i<10; i++){
        bus0.count();                 /* 評価対象となるバスに乗る変数にデータ書き込みが */
        a=i;                          /* 行われた場合、直前に評価関数を埋め込む */

        bus1.count_bit(32, 8);
        b=i+1;

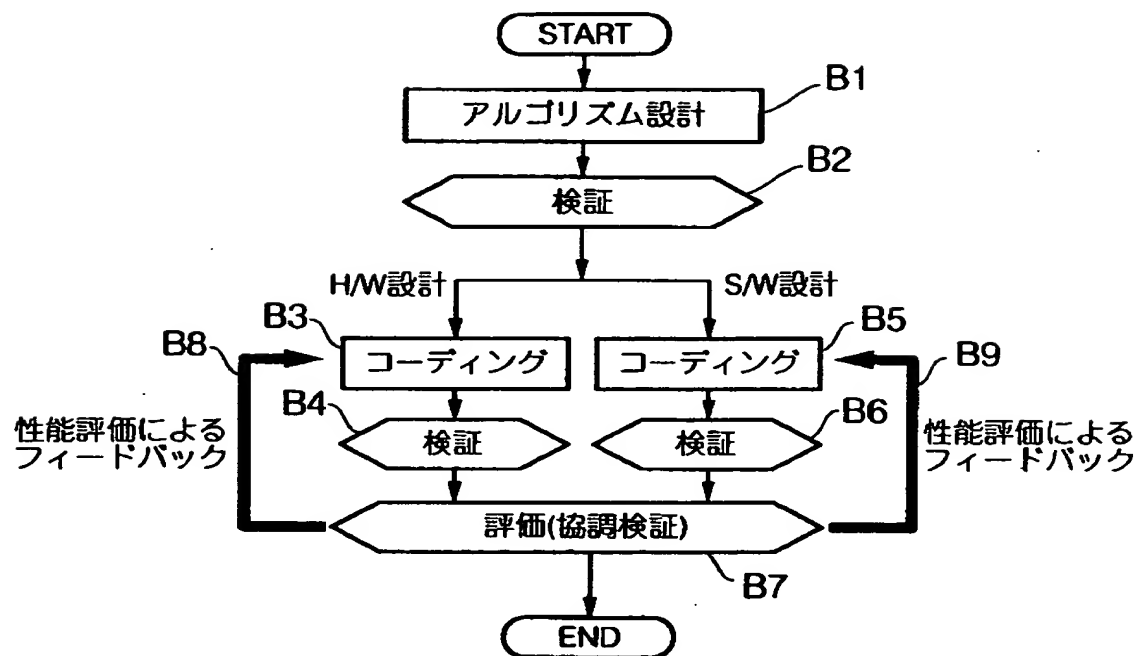
        bus2.count();
        c[i]=i*2;
    }

    bus0.print();
    bus1.print();
    bus2.print();

    return 0;
}

```

【図 1 1】



【書類名】 要約書

【要約】

【課題】 コーディング後の協調検証におけるハードウェア、ソフトウェアの切り分けに関するフィードバックループを排し、設計ターンアラウンドタイムの大幅な削減を行う。

【解決手段】 アルゴリズム設計で作成したC言語やC⁺⁺等汎用高級言語で記述されたソースをハードウェア、ソフトウェアに切り分けた後（ステップA3）、バスのトラフィックをカウントする評価関数を作成して（ステップA4）、バスにデータ(変数)が乗る毎に、その評価関数を実行するようにソースを修正し（ステップA6）、その後、性能評価を行い（ステップA7）、最終的に算出された処理レートに対するバストラフィック（ステップA8、A9）により、アーキテクチャー設計段階でハードウェア、ソフトウェアの切り分けを最適に行う（ステップA16）。

【選択図】 図1

認定・付加情報

特許出願の番号	特願2000-166630
受付番号	50000689803
書類名	特許願
担当官	塩崎 博子 1606
作成日	平成12年 6月 9日

<認定情報・付加情報>

【特許出願人】

【識別番号】	000232036
【住所又は居所】	神奈川県川崎市中原区小杉町1丁目403番53
【氏名又は名称】	日本電気アイシーマイコンシステム株式会社

【代理人】

申請人

【識別番号】	100108578
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所

【氏名又は名称】	高橋 詔男
----------	-------

【代理人】

【識別番号】	100064908
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所

【氏名又は名称】	志賀 正武
----------	-------

【選任した代理人】

【識別番号】	100101465
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所

【氏名又は名称】	青山 正和
----------	-------

【選任した代理人】

【識別番号】	100108453
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所

【氏名又は名称】	村山 靖彦
----------	-------

次頁無

出 願 人 履 歴 情 報

識別番号 [000232036]

1. 変更年月日 1990年 8月13日

[変更理由] 新規登録

住 所 神奈川県川崎市中原区小杉町1丁目403番53

氏 名 日本電気アイシーマイコンシステム株式会社